[1] What are Requirements?

A **requirement** is a statement that identifies a capability or function that is needed by a system to satisfy customer needs.

Requirements are identified by names such as REQ01.

Each requirement must exhibit the following characteristics

1. It is **verifiable**, meaning it can be proven to be present in the system through the use of tests, inspection, analysis etc.
2. It is **necessary,** meaning it directly supports the overall objectives of the system.
3. It is **attainable**
4. It **Expresses Need,** meaning it states WHAT is needed, not HOW to accomplish it
5. It must be **complete,** meaning the requirement alone should provide the details necessary to implement it. If applicable it should include specific conditions, use numbers and units, mention specific parts of the system/software etc. For example see section 3.2.

Careful consideration must be given to the introduction of each requirement as unnecessary requirements add unjustified cost to the system. Thus each requirement must introduce something to the system that cannot be fulfilled by any other requirement.

[2] Levels of Requirements

Requirements can be divided into levels by their specificity. We will categorize requirements using the following levels

1. **Client**
   - Highest level = least specific

- plain english, what is the objective of the entire system (what are we doing?)

2. **System**

   - The environment our software must perform in

   - Specify minimum conditions that must be present to allow for the successful execution of our software.

   - Must be validated before the design phase begins (before we work with code)

3. **Software**

   - From the moment our product is run until the moment it is terminated, detail what tasks must be performed to satisfy all client level requirements

[3] <u>Validation</u>

In this step we confirm the completeness and correctness of requirements and eliminate any requirement that does not satisfy all characteristics outlined in section 1.

To test if a requirement is necessary, ask *"what is the worst that could happen if we removed it?"*

Confirm that all Software level requirements are possible in the environment specified by the System level requirements.

Ensure no requirement contradicts another requirement.

[3.1] Carefully consider the specifics of each requirement and ask if it is even practical to apply this requirement. For example consider the following…

A new manned aircraft has the following, seemingly valid requirement:

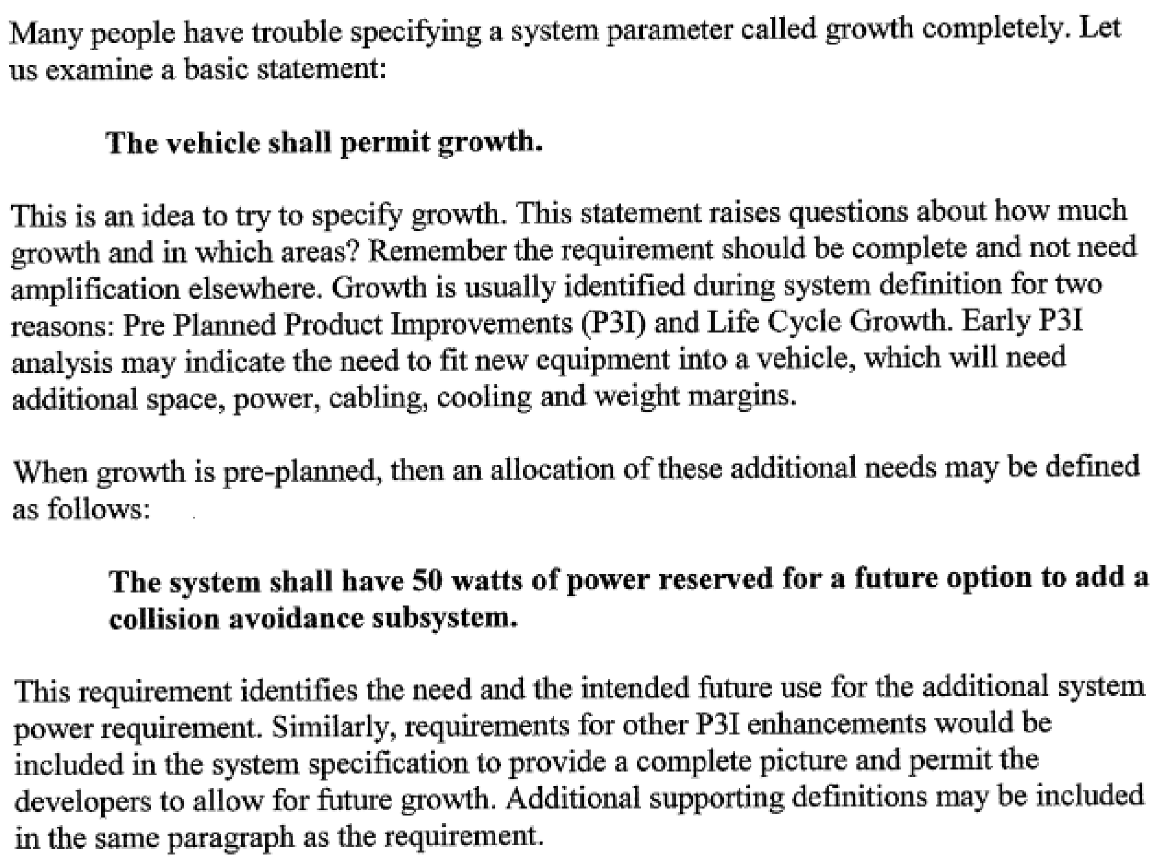
The aircraft shall withstand a 16G turn for at least 30 seconds.

However, after looking at the operational environment, we find that a pilot will black out somewhere between 7-10Gs. Further, at 10Gs, blackout typically occurs in a second or two. The validity of the above requirement then becomes suspect and the implementation may lead to gold-plating at least, and expensive, unnecessary testing and rework at the worst. Gold plating is the application of requirements that overstate the need for the system or at a level higher than is useful to apply. A more realistic requirement may be:

The aircraft shall withstand a 12G turn for at least 15 seconds.

The resulting validated requirement not only is less costly to achieve, but will support the operational environment with adequate margin. In this case, a separate discussion on the inherent margin would supplement the requirement. Providing relevant background information can support further analysis and requirement evolution. The result of validating requirements allows the developer to create the right system.

[3.2] Check completeness. Consider the following…



Many people have trouble specifying a system parameter called growth completely. Let us examine a basic statement:

The vehicle shall permit growth.

This is an idea to try to specify growth. This statement raises questions about how much growth and in which areas? Remember the requirement should be complete and not need amplification elsewhere. Growth is usually identified during system definition for two reasons: Pre Planned Product Improvements (P3I) and Life Cycle Growth. Early P3I analysis may indicate the need to fit new equipment into a vehicle, which will need additional space, power, cabling, cooling and weight margins.

When growth is pre-planned, then an allocation of these additional needs may be defined as follows:

The system shall have 50 watts of power reserved for a future option to add a collision avoidance subsystem.

This requirement identifies the need and the intended future use for the additional system power requirement. Similarly, requirements for other P3I enhancements would be included in the system specification to provide a complete picture and permit the developers to allow for future growth. Additional supporting definitions may be included in the same paragraph as the requirement.

[4] <u>Terminology</u>

Requirements use **shall**

- The [system / subsystem / element] shall …

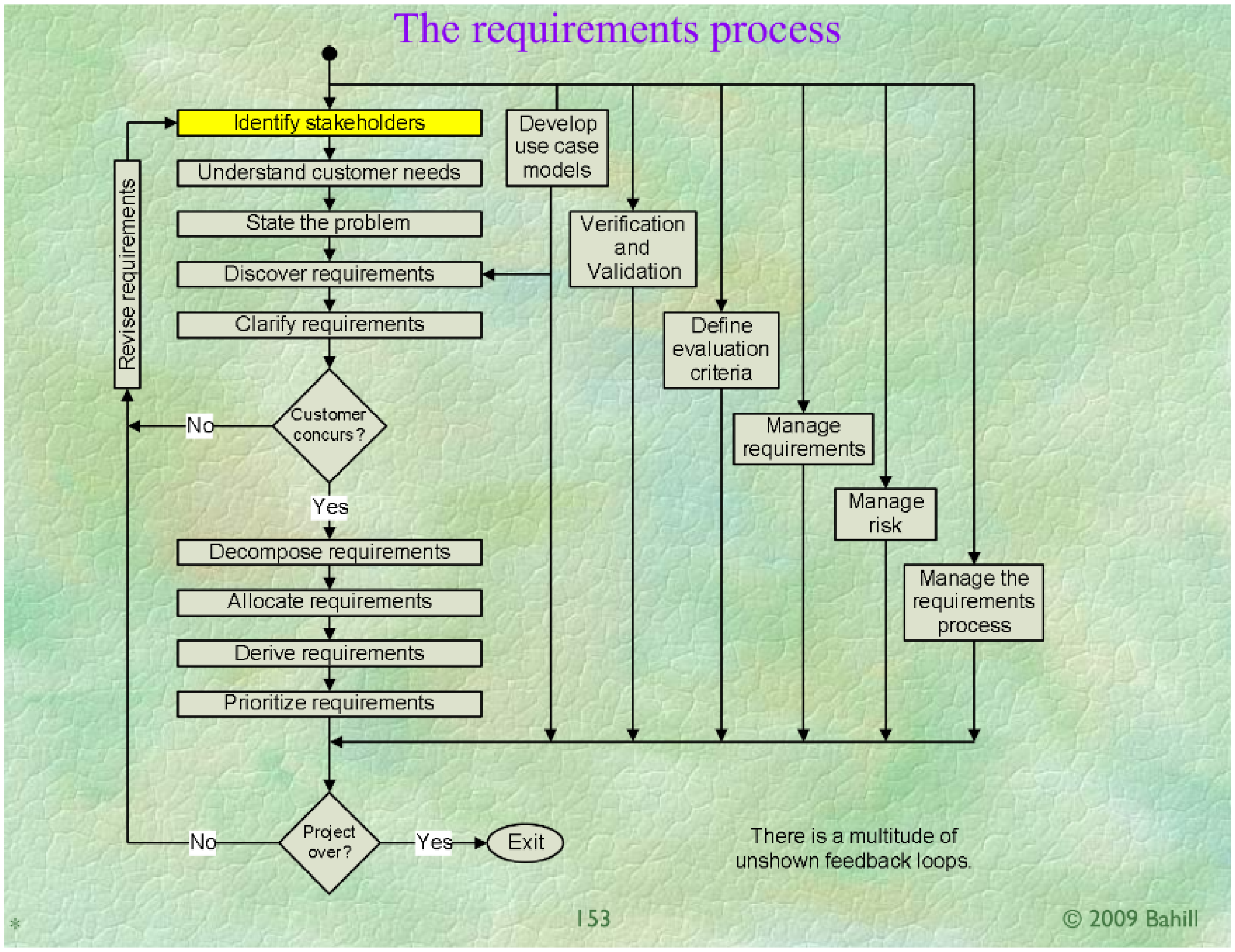
Goals are special requirements that are not imperative, goals use **should**

- The [system / subsystem / element] should … [some specific goal]

Statements of fact use **will**, these are not considered requirements

- The [system / subsystem / element] will … [Some observation]

[5] Timeline



[6] Requirements checklist

After writing each requirement go through the following list:

- This checklist may include the following questions:
  - Is the requirement clear?
  - Is the requirement correct?
  - Are there conflicts with other requirements?
  - Is the requirement achievable?
  - Is the requirement necessary?
  - Does the requirement specify a solution?
  - Is the requirement verifiable?
  - Is the requirement traceable?

- Is the requirement set complete?